

This application is submitted in the names of inventor Steve W. Brown, assignor to Zayante, Inc., a California Corporation.

5
10
SPECIFICATION

15
20
A METHOD AND APPARATUS FOR SUPPORTING AND PRESENTING
MULTIPLE SERIAL BUS NODES USING
DISTINCT CONFIGURATION ROM IMAGES

25
BACKGROUND OF THE INVENTION

1. Field of the Invention

30
This invention pertains generally to configuration ROM implementations for IEEE Standard 1394 nodes. More particularly, the invention is a method and apparatus for presenting a plurality of link devices as separate nodes within a single serial bus module by generating individual or a distinct configuration ROM image for each link device in the module.

35 2. The Prior Art

The Institute of Electrical and Electronics Engineers, Inc. (IEEE) defines the IEEE Standard 1394-1995 serial bus architecture in the document "IEEE

Standard for a High Performance Serial Bus” published August 30, 1996 which is incorporated herein by reference. In IEEE 1394, the serial bus architecture is defined in terms of nodes. In general, a node is an addressable entity (i.e., a logical entity with a unique address), which can be independently reset and identified. More than one node may reside on a single module, and more than one unit may reside in a single node.

A module is a physical device, comprising one or more nodes that share a physical interface. The address space provided by a node can be directly mapped to one or more units. A unit is a logical entity, such as a disk controller, which corresponds to unique I/O (input/output) driver software. On a multifunction node, for example, a processor and I/O interfaces could be different units on the same node.

Modules and/or nodes can be “interconnected” with each other using an appropriate physical topology suitable for use with the serial bus, such as a “backplane environment” and/or “cable environment”, for example. These environments are described in further detail in Institute of Electrical and Electronics Engineers (IEEE) Standard 1394-1995 “IEEE Standard for a High Performance Serial Bus” published August 30, 1996. Interconnected nodes may reside in either environment without restriction.

Configuration ROM implementations are well known in the field of serial bus devices and provide the hardware and software specifications of a serial bus node and its associated units. For example in IEEE Standard 1394, two

configuration ROM formats are supported: minimal and general. The minimal ROM format provides a 24-bit company identifier. The general ROM format provides additional information in a `bus_info_block` and a `root_directory`.

Entries within the `root_directory` may provide information or may provide a
 5 pointer to another directory (root-dependent directory and/or `unit_directory`),
 which has the same structure as the `root_directory`. Entries within the root
 directory may also provide a pointer to a leaf, which contains information. The
`unit_directories` contain information about the units associated with the node,
 such as their software version number and their location within the address space
 10 of the node, for example.

FIG. 1 shows a general ROM implementation format for IEEE Standard 1394. The ROM directory structure is a hierarchy of information blocks, where the blocks higher in the hierarchy point to the blocks beneath them. The location
 15 of the initial blocks (`info_length`, `crc_length`, `rom_crc_value`, `bus_info_block`, and `root_directory`) are fixed. The location of the other entries (`unit_directories`, root and unit leaves) varies according to each vendor, but are specified by entries within the `root_directory` or its associated directories.

20 In general, the `bus_info_block` provides specific information about the node. For example, the `bus_info_block` may indicate whether the node carries out isochronous data transfers. Additionally, the `bus_info_block` provides a `node_vendor_id` field, a `chip_id_hi` field, and a `chip_id_lo` field, among other things. Together, the `node_vendor_id`, `chip_id_hi`, and `chip_id_lo` fields forms a
 25 64-bit node unique identifier. Other node specific information may be provided in

the root_directory and the root leaves of the ROM. Unit specific information is normally provided in the unit_directory and the unit leaves of the ROM. For example, the specification identification and the version number may be provided for a particular protocol in the unit_directory and the unit leaves. IEEE Standard 5 1394-1995 "IEEE Standard for a High Performance Serial Bus" published August 30, 1996 describes the general ROM format and its associated blocks in further detail and is incorporated herein by reference.

According to the prior art, a serial bus module may include one or more 10 nodes. For example, FIG. 2 illustrates a typical module device 1 having first and second nodes 2a, 2b. Nodes 2a, 2b include respective link layer services (LINK) 3a, 3b and physical layer services (PHY) 4a, 4b. Each link device 3a, 3b includes a respective global unique identifier (GUID) 5a, 5b to identify each node device 2a, 2b.

15 Presently, the configuration ROM described above is managed by software operating at the transaction layer 6 in module 1. However, current transaction layer implementations which support multiple link devices (such as depicted in FIG. 2) present a single configuration ROM image 7 for both link 20 devices. As a result, transaction layer software 6 presents nodes 2a and 2b as the same GUID, which may result in inconsistent information provided to the serial bus 8.

Other node or module devices (not shown) attached to serial bus 8 may 25 query module 1 to ascertain certain configuration data associated with module 1.

For example, a remote node may query module 1 to ascertain, among other things, the node configuration of module 1 and/or the units presented by the nodes of module 1. These remote nodes query module 1 using one of various request commands. Some remote nodes request information "by quadlet" and other
5 nodes request information "by block", for example.

When a request is made by quadlet, the corresponding link device 3a, 3b provides the requested data from the hardware registers 9a, 9b associated with the respective link device 3a, 3b. In this manner, link device 3a provides the
10 requested data from its hardware registers 9a, and link device 3b provides the requested data from its hardware registers 9b.

However, when a request is made by block, the requested data is provided from the configuration ROM 6 which is normally managed by the transaction
15 layer software 5. As noted above, present transaction layer implementations provide a single configuration ROM 6 for multiple link devices 3a, 3b. Thus the data provided in conjunction with a request by block may be different and inconsistent with that provided had the request been made by quadlet.

20 Accordingly, there is a need for a method for presenting a plurality of link devices as separate nodes within a single serial bus module by generating an individual or distinct configuration ROM image for each link device in the module so that when a request is made to the module, accurate and consistent data is provided to the requesting device. The present invention satisfies these

needs, as well as others, and generally overcomes the deficiencies found in the background art.

An object of the invention is to provide a method for supporting multiple
5 link devices in a single module which overcomes the deficiencies of the prior art.

Another object of the invention is to provide a method for presenting a plurality of link devices as separate nodes within a single serial bus by providing an individual configuration ROM for each link device in the module.

10

Further objects and advantages of the invention will be brought out in the following portions of the specification, wherein the detailed description is for the purpose of fully disclosing the preferred embodiment of the invention without placing limitations thereon.

15

BRIEF DESCRIPTION OF THE INVENTION

The present invention is a method and apparatus embodied in transaction layer software suitable for use with serial bus devices, such as IEEE standard
20 1394 serial bus devices. The invention further relates to machine readable media on which are stored embodiments of the present invention. It is contemplated that any media suitable for retrieving instructions is within the scope of the present invention. By way of example, such media may take the form of magnetic, optical, or semiconductor media. The invention also relates to data

structures that contain embodiments of the present invention, and to the transmission of data structures containing embodiments of the present invention.

In its most general terms, the invention comprises software for supporting multiple link devices in the same physical module as separate nodes by presenting individual or distinct configuration ROMs for each link device to the serial bus. In the preferred embodiment, the software comprises IEEE standard 1394 transaction layer software (TNF kernel) for a serial bus module attachable to an IEEE standard 1394 bus. The software is executed by a conventional processor within the module device as is known in the art.

The serial bus module may include one or more link devices operatively coupled for communication with the TNF kernel. In other exemplary embodiments, device driver services may be used to manage communication between the TNF kernel and the link devices as is known in the art.

The TNF kernel carries out the operation of ascertaining or becoming aware of the link devices in the modules, creating an individual configuration ROM for each link device, and presenting the created configuration ROMs to the other devices on the 1394 bus to thereby present each link device in the module as a separate node.

The TNF kernel ascertains each link device normally during initialization of the module, either through a predefined startup routine or through notification from device driver services, if such services are implemented in the module. The

TNF kernel ascertains, among other things, each link device's GUID (globally unique identifier).

For each link device, the TNF kernel creates a data structure suitable for storing data associated with each link device. In an exemplary embodiment of the invention, the TNF kernel uses a data structure comprising a list of link data records, one record for each link device. Each link data record includes a CSR (control and status register) address map and Configuration ROM image storage and/or support thereof. Each Configuration ROM image is constructed using, among other things, the information for each link device and includes an entry for the link device GUID. The CSR address map is a data structure which among other things, points to the active configuration ROM. In one of a number of possible embodiments, the present invention may employ dynamic configuration ROM using double image buffers which is described in copending application having the title "A SYSTEM AND METHOD FOR PROVIDING DYNAMIC CONFIGURATION ROM USING DOUBLE IMAGE BUFFERS" and the attorney docket number "ZAY-99-029" filed on November 1, 1999 and is incorporated herein by reference.

Requests by other devices to the module are communicated from the serial bus to the physical layer device in the present module. Communications for layers higher than the physical layer are then communicated to the link layer device for further processing.

5 However, when a request by block is made to the module, the request is communicated from the corresponding link device to the TNF kernel. In addition, the link device provides its unique software ID (Link ID) along with the request. In response to the block request, the TNF kernel provides the configuration ROM for the appropriate link device according to the Link ID. Since individual
10 configuration ROMs are created for each link device in the module, the information provided by the TNF kernel via the individual configuration ROM is consistent with information provided in response to requests made by quadlet as described above.

15

20

25

25

FIG. 3 is a functional block diagram of an illustrative embodiment serial device module which carries out multiple link device presentation according to the present invention.

5

FIG. 4 is a functional block diagram of an illustrative communication system which includes a serial device module configured to carry out multiple link device presentation according to the present invention.

FIG. 5 is a flow chart showing generally acts for supporting and presenting a plurality of separate nodes as separate nodes according to the present invention.

15

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Persons of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to such skilled persons having the benefit of this disclosure.

Referring more specifically to the drawings, for illustrative purposes the present invention is embodied in the apparatus shown FIG. 3 and FIG. 4 and the method outlined in FIG. 5. It will be appreciated that the apparatus may vary as to configuration and as to details of the parts, and that the method may vary as to

details and the order of the acts, without departing from the basic concepts as disclosed herein. The invention is disclosed generally in terms of a method and apparatus for use with IEEE standard 1394 devices, although numerous other uses for the invention will suggest themselves to persons of ordinary skill in the art.

Referring now to FIG. 3, there is shown a functional block diagram of an illustrative embodiment serial device module 10 which carries out multiple link device support and presentation according to the present invention. Module 10 includes two nodes 12a, 12b, each having a respective link layer (LINK) device 14a, 14b connected to a respective physical layer (PHY) device 16a, 16b. LINKS 14a, 14b provide the link services for the module 26 as is known in the art, and PHY devices 16a, 16b provide the physical layer services for the module 10 as is known in the art. Each PHY device 16a, 16b is connected to serial bus 18 through a conventional serial interface connection, such as cables, connectors and physical silicon, for example.

The module 10 further includes one or more unit architectures 20 to present to other devices on the serial bus 18. Unit architectures 20 may comprise conventional units, such as a disk controller or some other storage device and a scanner controller, for example. These unit architectures 20 are presented to the serial bus by the generated configuration ROM as described further below and described above in conjunction with FIG. 1.

5 The unit architectures 20 and the LINKS 14a, 14b are operatively coupled for communication to TNF kernel 22. The TNF kernel 22 provides transactional services for module 10 and the method of the invention as described herein and in further detail in conjunction with FIG. 5. It will be appreciated that module 10 is only exemplary, and other arrangements of may be utilized without departing from the spirit and scope of the present invention. For example, the invention may be used when one or more link devices are present. Module 10 is depicted with two link devices for illustrative purposes only. Additionally, as noted above, a device driver service may be used to facilitate communication between the TNF
10 kernel 22 and the LINKS 14a, 14b without departing from the scope of the invention.

Each of the LINKs 14a, 14b further includes a respective global unique identifier (GUID) 24a, 24b which identifies each node 12a, 12b to other nodes on
15 the serial bus.

The TNF kernel 22 becomes aware of each link device normally during initialization of the module 10, either through a predefined startup routine or through notification from device driver services, if such services are implemented
20 in the module. The TNF kernel 22 ascertains, among other things, each link device's GUID 24a, 24b.

The TNF kernel 22 then creates, normally within RAM (not shown), a data structure for storing configuration ROM data for each link device 14a, 14b.
25 Various data structures may be used for storing configuration ROM data,

however, in the preferred embodiment, the TNF kernel 22 uses a linked list of “link data records”, one data record for each link device 14a, 14b. Thus, TNF kernel 22 creates a configuration ROM 26a, 26b corresponding to each link device 14a, 14b. For each configuration ROM 26a, 26b, the TNF kernel 22 creates
 5 a GUID entry for the corresponding link device 14a, 14b. Thus, ROM 22a which is associated with link 14a includes an entry for GUID1 24a, and ROM 22b which is associated with link 14b includes an entry for GUID1 24b. With this arrangement, a one-to-one mapping is made with LINK 14a to node 12a via configuration ROM 26a, and with LINK 14b to node 12b via configuration ROM
 10 26b.

The CSR address map of node 10 includes pointers to various items associated with node 10, including the ROM associated with each link device 14a, 14b.

15 As noted above, the present invention may be used in conjunction with the dynamic configuration ROM implementation described in copending application having the title “A SYSTEM AND METHOD FOR PROVIDING DYNAMIC CONFIGURATION ROM USING DOUBLE IMAGE BUFFERS” and
 20 the attorney docket number “ZAY-99-029” filed on November 1, 1999 and is incorporated herein by reference. This embodiment is one of a number of possible embodiments.

Referring now to FIG.4, as well as FIG. 3, there is generally shown a
 25 functional block diagram of an illustrative communication system 30 which

includes a serial device module configured to carry out multiple link device support and presentation according to the present invention.

System 30 includes a module 10 structured and configured as described
5 above in conjunction with FIG. 3. Module 10 includes nodes 12a, 12b
represented by PHY 16a operatively coupled to LINK 14a and PHY 16b
operatively coupled to LINK 14b. LINKS 14a, 14b are coupled for
communication to TNF kernel 22 (although a device driver service may also be
used). As depicted in FIG. 4, TNF kernel 22 has created configuration ROM 26a
10 associated with link 14a and ROM 26b associated with link 14b as described
above.

Module 10 is connected to the serial bus 18 via lines 28a and 28b, where
line 28a is coupled to PHY 16a (Node 12a) and line 28b is coupled to PHY 16b
15 (Node 12b). As depicted in FIG. 4, Node 12a is configured for 1394 IP (Internet
Protocol) communication, while Node 12b is configured for 1394 AV/C
(Audio/Video Control) communication. Thus, IP requests to Module 10 are
communicated through line 28a, then through PHY 16a, then through LINK 14a
to TNF kernel 22. AV/C requests to Module 10 are communicated through line
20 28b, then through PHY 16b, then through LINK 14b to TNF kernel 22. When
such requests are routed from the LINKS 14a, 14b to TNF kernel 22, the LINK
devices also communicate a unique software ID (Link ID) to thereby indicated
which link device is passing the request.

Module 10 further includes Nodes 30a, 30b each operatively coupled to the serial bus 18. Nodes 30a, 30b may comprise any serial bus device capable of communication with Module 10. Other modules having a plurality of nodes (not shown) may also be connected to the serial bus 18 and communicate with

5 Module 10 in substantially the same manner as described herein with respect to nodes 30a, 30b. As shown, node 30a is structured and configured for 1394 IP communication, while node 30b is structured and configured for 1394 AV/C communication.

10 In operation, when node 30a makes a block read request (1394 IP) to module 10, such request is communicated along bus 18 and line 28a to PHY 16a. PHY 16a then communicates the request to LINK 14a, for further processing. LINK 14a then communicates the request to TNF kernel 22, along with Link ID of LINK 14a. In response, TNF kernel 22 provides the configuration ROM
15 information associated with LINK 14a from ROM 26a. As noted above, ROM 26a includes the GUID associated with LINK 14a. This configuration ROM information is then communicated back to Node 30a via LINK 14a, then through PHY 16a, line 28a, and bus 18 ultimately to the requesting node 30a.

20 In contrast to the previous data path outlined above for node 30a , when node 30b makes a block read request (1394 AV/C) to module 10, such request is communicated along bus 18 and line 28b to PHY 16b. PHY 16b then communicates the request to LINK 14b, for further processing. LINK 14b then communicates the request to TNF kernel 22, along with Link ID of LINK 14b. In
25 response, TNF kernel 22 provides the configuration ROM information associated

with LINK 14b from ROM 26b, which includes the GUID associated with LINK 14b. This configuration ROM information is then communicated back to Node 30b via LINK 14b, then through PHY 16b, line 28b, and bus 18 ultimately to the requesting node 30b.

5

As illustrated above, TNF kernel 22 provides the corresponding configuration ROM (including correct GUID data) for the link device in which the request is carried out and in which communication is carried through. As such, a consistent one to one mapping between link devices and nodes are provided within the multiple-link device module (in the above example module 10) even when the request is a request "by block".

The method and operation of the invention will be more fully understood by reference to the flow chart of FIG. 5, as well as FIG. 3 and FIG. 4. FIG. 5 illustrates generally the actions associated with supporting and presenting a plurality of separate nodes as separate nodes according to the present invention. The order of operation as shown in FIG. 4 and described below are only exemplary, and should not be considered limiting.

At box 100, the TNF kernel 22 operating in module 10 becomes aware of the link devices in the module. As noted above, this is carried out normally during initialization of the module 10, either through a predefined startup routine or through notification from device driver. The TNF kernel 22 ascertains, among other things, each link device's GUID 24a, 24b. Box 110 is then carried out.

25

At box 110, the TNF kernel 22 creates or otherwise generates a configuration ROM image for each link device within the module 10. Thus, in FIG. 3 and FIG. 4, the TNF kernel 22 creates configuration ROM 26a for link 14a and ROM 26b for link 14b. Each configuration ROM 26a, 26b will include such ROM information related to the respective LINK 14a, 14b including an entry for the respective GUID 24a, 24b. The format of the configuration ROM 26a, 26b complies to the standard set forth for the device which in the present example is IEEE standard 1394 as described above in conjunction with FIG. 1. Box 120 is then carried out.

10

At box 120, the module 10 receives a block request from another node (or module) on the serial bus. The request is received in respective PHY 16a, 16b. Which PHY receives the request varies according to various factors, including which protocol is supported, for example as illustrated in FIG. 4. The PHY then communicates the request to the LINK. The LINK in turn communicates the request to the TNF kernel 22 along with the LINK ID. As noted above, the LINK ID is used by the TNF kernel 22 to ascertain which configuration ROM 26a, 26b to provide to the requesting node. BOX 130 is then carried out.

15

20

At box 130, the TNF kernel 22 determines which configuration ROM 26a, 26b is requested according the LINK ID communicated from the LINK device from box 120. The TNF kernel 22 then provides the configuration ROM associated with the LINK ID to the requesting node. If the module uses the dynamic configuration ROM using double image buffers, noted above, then the

TNF kernel provides the “active” ROM image according the LINK ID provided.
Step 120 may then be carried out again for additional block requests.

Accordingly, it will be seen that this invention provides a method which
5 supporting multiple link devices in the same physical module as separate nodes
by presenting individual or distinct configuration ROMs for each link device to
the serial bus. Although the description above contains many specificities, these
should not be construed as limiting the scope of the invention but as merely
providing an illustration of the presently preferred embodiment of the invention.
10 Thus the scope of this invention should be determined by the appended claims
and their legal equivalents.

662077-6982460